Acquisition of Entity-Relationship Models from Natural Language Specifications using Heuristics

Nazlia Omar[1], Paul Hanna[2] and Paul Mc Kevitt[3]

[1] Department of Computer Science, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Email: no@ftsm.ukm.my

[2] School of Computing and Mathematics, Faculty of Engineering, Jordanstown Campus, University of Ulster, Newtownabbey BT37 0QB, Northern Ireland, UK

[3] School of Computing and Intelligent Systems, Faculty of Engineering, Magee College, University of Ulster, Derry/Londonderry BT48 7JL, Northern Ireland, UK

ABSTRACT

This paper describes a heuristics-based approach in the semi-automated generation of Entity-Relationship (ER) diagrams for database modelling from natural language specifications and describes the implementation of such a system called *ER-Converter*. Though this is a semi-automatic transformation process, ER-Converter aims to require minimal human intervention during the process. ER-Converter has been evaluated in blind trials against a set of database problems. ER-Converter has an average of 90% recall and 85% precision. In terms of user intervention, ER-Converter requires very little human assistance with only 1.6% in the test dataset. The evaluation results are discussed and demonstrate that ER-Converter could be used, for example, within the domain model of a multimedia intelligent tutoring system, designed to assist in the learning and teaching of databases.

Keywords: heuristics, entity-relationship, natural language processing

INTRODUCTION

Entity-relationship modelling can be a daunting task to both students and designers alike due to its abstract nature and technicality. Much research has attempted to apply natural language processing in extracting knowledge from requirements specifications with the aim to design databases. However, research on the formation and use of heuristics to aid the construction of logical databases from natural language has been scarce.

This paper describes the development of a tool, ER-Converter, which transforms a natural language text input into an ER model. This is a heuristics-based tool which employs syntactic heuristics during the transformation. In order to achieve the desired result, new and existing heuristics are applied during the process. Though this is a semi-automatic transformation process, the tool aims to provide minimal human intervention during the process.

BACKGROUND AND PREVIOUS WORK

This section provides a brief summary on data modelling which introduces the concept of ER Model and reviews the previous work that applies natural language processing to Databases. The existing tools, techniques and limitations are discussed. Some of the work like DMG[10] provides a basis for the development of new heuristics applied in ER-Converter.

Overview of Data Modelling

The first step in designing a database application is to understand what information the database must store. This step is known as requirements analysis. The information gathered in this step is used to develop a high-level description of the data to be stored in the database. This step is referred to as conceptual design, and it is often carried out using the ER model. ER models are built around the basic concepts of *entities*, *attributes, relationships* and *cardinality.* An *entity* is an object that exists in the real world and is distinguishable from other objects. These are typically derived from nouns. Examples of entities include the following: a "student", an "employee" and a "book". A collection of similar entities is called an *entity set.* An entity is described using a set of *attributes*. The attributes of an entity reflect the level of detail at which we wish to represent information about

entities. Attributes may be derived from adjectives and adverbs. For example, the "Student" entity set may have "ID_number", "Name", "Address", "Course" and "Year" as its attributes. A *relationship* is an association among two or more entities. Relationships can be typically derived from verbs. For example, we may have a relationship from this sentence: A student may "take" many courses. "take" implies a relationship between the entity "student" and "course". *Cardinality* represents the key constraint in a relationship. In the previous example, the cardinality is said to be many-to-many, to indicate that a student can take many courses and a course can be taken by many students. In an ER diagram, an entity is normally represented by a rectangle. An ellipse usually represents an attribute meanwhile a diamond shape shows a relationship. Cardinality is represented by 1 for the one-sided and M for the many-sided.

Applying Natural Language Processing (NLP) to Databases

Much work [2,5,6,10] has attempted to apply natural language in extracting knowledge from requirements specifications or dialogue sessions with designers with the aim to design databases. Dialogue tool [2] is a knowledge-based tool applied to the German language for producing a skeleton diagram of an Enhanced Entity-Relationship (EER) model. This tool is part of a larger database design system known as RADD (Rapid Application and Database Development) which consists of other components that form a complex tool. In order to obtain knowledge from the designer, a moderated dialogue is established during the design process. The transformation of the structure of natural language sentences into EER model structures is a process which is based on heuristic assumptions and pragmatic interpretation. The aim of the pragmatic interpretation is the mapping of the natural language input onto EER model structures using the results of the syntactic and semantic analyses. One major limitation in this system is that the accuracy of the EER model produced depends on the size and complexity of the grammar used and the scope of lexicon.

ANNAPURNA [5] is project aimed to provide a computerized environment for semi-automatic database design from knowledge acquisition up to generating an optimal database schema for a given database management system. ANNAPURNA concentrated on the phases concerned with acquiring the terminological rules. The first step in acquisition of the terminological knowledge involves extracting the knowledge from queries and rules that have the form of natural language expressions. The knowledge obtained would then be put into the form of S-diagrams. An S-diagram is a graphical data model which can be used to specify *classes* (for example room and door), subclass connections between classes (for example rooms and doors are physical objects) and attributes. The limitation of the above work is that the use of S-diagrams performs best when the complexity is small.

DMG [10] is a rule based design tool which maintains rules and heuristics in several knowledge bases. A parsing algorithm which accesses information of a grammar and a lexicon is designed to meet the requirements of the tool. During the parsing phase, the sentence is parsed by retrieving necessary information from the grammar, represented by syntactic rules and the lexicon. The parsing results are processed further on by rules and heuristics which set up a relationship between linguistic and design knowledge. The DMG has to interact with the user if a word does not exist in the lexicon or the input of the mapping rules is ambiguous. The linguistic structures are then transformed by heuristics into EER concepts. Though DMG proposed a large number of heuristics to be used in the transformation from natural language to EER models, the tool has not yet been developed into a practical system.

E-R generator [6] is another rule-based system that generates E-R models from natural language specifications. The E-R generator consists of two kinds of rules: specific rules linked to semantics of some words in sentences, and generic rules that identify entities and relationships on the basis of the logical form of the sentence and on the basis of the entities and relationships under construction. The knowledge representation structures are constructed by a natural language understander (NLU) system which uses a semantic interpretation approach. There are situations in which the system needs assistance from the user in order to resolve ambiguities such as the attachment of attributes and resolving anaphoric references.

CM-Builder [8] is a natural language based CASE tool which aims at supporting the analysis stage of software development in an object-oriented framework. The tool uses natural language processing techniques to analyse software requirements documents and produces initial conceptual models represented in Unified Modelling Language. The system uses discourse interpretation and frequency analysis in producing the conceptual models. CM-Builder still has some limitation in the linguistic analysis. For example, attachment of postmodifiers such as prepositional phrases and relative clauses is limited. Other shortcomings include the state of the knowledge bases which are static and not easily updateable nor adaptive.

All the systems discussed here have user involvement during processing. Because of the incomplete presentation of knowledge, ambiguities and redundancies, full automation of the design process is fundamentally impossible [5]. As a consequence, the tools must be able to interact with the designer, including ER-Converter. A semi-automatic design process is far more economical than an entirely manual process [5].

HEURISTICS TO IDENTIFY ER ELEMENTS

Heuristics represent an indefinite assumption [10], often guided by common sense, to provide good but not necessarily optimal solutions to difficult problems, easily and quickly [11]. Research on the formation and use of heuristics to aid the construction of logical database structures from natural language has been scarce. The only existing work that proposes a large number of heuristics to be used in the transformation from natural language to ER models is DMG [10]. However the work has not been implemented. The authors of DMG proposed both syntactic and semantic heuristics to be applied in extracting knowledge from requirements specifications. Although E-R Generator [6] and RADD [2] utilized heuristics in their work, they do not detail a precise set of heuristics in their approach. Chen [3] suggested that the basic constructs of English sentences could be mapped into ER schemas in a natural way and presented a set of rules to put forward the ideas. Though the set is referred to as "rules", Chen mentioned that they are better viewed as "guidelines" as it is possible to find counter examples to them. Here we regard Chen's "rules" as heuristics as they are largely "rules-of-thumb" based on observations rather than theoretically derived. Only heuristics for language syntax are considered and proposed at this stage.

Here, a selection of the heuristics applied in the transformation from database specifications to the data modeling constructs is presented. These heuristics are gathered from past work [3,9,10] and some are newly formed. A total of 21 previously published and newly proposed heuristics were identified. Some examples in terms of sentences are provided to illustrate the application of heuristics which are context dependent.

Heuristics to determine entities:

1. Heuristic HE2: A common noun may indicate an entity type [3,10].
2. Heuristic HE3: A proper noun may indicate an entity [3,10].
3. Heuristic HE7: If consecutive nouns are present, check the last noun. If it is not one of the words in set S where S={number, no, code, date, type, volume, birth, id, address, name}, most likely it is an entity type. Else it may indicate an attribute type.

4. Heuristic HE8: If a noun occurs before the verb 'has'/ 'have', it may indicate an entity type. For relationships of the form A *have/has* B where A and B are both nouns, the occurrence of A may indicate that it is an entity. This is true in cases where the relationship between A and B implies B instance-of A or B component-of A. This is illustrated in the following example:
   "Each piece of equipment has an equipment number and a description".
   In this example, "equipment" may suggest that it is an entity type due to its occurrence prior to the "has" verb phrase.

Heuristics to exclude non-potential entity types candidates:

1. Heuristic HEX: A noun such as "record", "database", "company", "system", "information" and "organization" may not be a suitable candidate for an entity type. For example, "company" may indicate the business environment and should not be included as part of the entity types. Examples:
   a) "An insurance company wishes to create a database to keep track of its operations."
   b) "An organization purchases items from a number of suppliers."

Heuristics to determine attributes:

1. Heuristic HA1: A noun which takes the general form of TERM_SUFFIX such as noun_id, noun_no, noun_type or noun_number may indicate an attribute type [9]. A noun such as "person_id", "group_no", "room_type" and "vehicle_number" may indicate an attribute type. The TERM_SUFFIX representation is often used in database problems' specifications.
2. Heuristic HA6: Genitive case in the noun phrase may indicate an attributive function [10].
3. Heuristic HA8: If a noun is followed directly by another noun and the latter belongs to set S where S={number, no, code, date, type, volume, birth, id, address, name}, this may indicate that both words are an attribute. Else it is most likely to be an entity.

Heuristics to determine relationships:

1. Heuristic HR1: An adverb can indicate an attribute for relationship [3].
2. Heuristic HR2: A transitive verb can be a candidate for relationship type [3].
3. Heuristic HR4: A verb followed by a preposition such as "on", "in", "by" and "to" may indicate a relationship type. For example: "Persons work on projects." Other examples include "assigned to" and "managed by".

Heuristics to determine cardinalities:

1. Heuristic HC2: The adjective "many" or "any" may suggest a maximum cardinality. For example:
   a) "A surgeon can perform many operations."
   b) "Each diet may be made of any number of servings."
2. Heuristic HC3: A comparative adjective "more" followed by the preposition "than" and a cardinal number may indicate the degree of the cardinality

between two entities. For example: "Each patient could have more than one operation."

Heuristics' Weights

The heuristics' weights are assigned according to the confidence level that the event is true. For example, HE2 (one of the heuristics to determine entity type) states that a common noun may indicate an entity type. It has been given a weight of 0.5. This basically means that 50% of the time this heuristic may produce the correct result, as not all nouns are entity types. Though the assignment of the weights is mainly based on intuition, these weights are also compared and reflected against the results obtained from training set.

Most of the values assigned lie between –1 and 1 with the exception of HEX which is assigned a value of 100. This value acts as a safe border that differentiates between an entity type and a non-entity type. For example, there may be much evidence occurring for a word indicating it is an entity type. This is reflected in the total sum of the weights of evidence found. As both entity types and non-entity types have positive values, a value of 100 and over may indicate strongly that a word may suggest a non-entity type. For attributes, all of the weights are assigned with negative values. The negative weights are assigned such that if more than one heuristic from either the entity or attribute type categories are applied to a word, this would reduce the sum of the total weights. The sum of weights can be outside of –1 and +1 range. Values approaching zero are treated as "low confidence". Two or more "weak" pieces of evidence are combined to give the weight an acceptable level of confidence. If this value falls within a threshold of –0.2 and 0.4, user intervention may be required to help identify its identity. The user will be prompted to decide whether the noun is an entity or an attribute. This is the only point where user intervention is needed in the process of generating the ER modelling concepts.

Training set

In order to test the newly developed heuristics, a manual test was carried out prior to the implementation of ER-Converter. This stage is seen as an important phase as the heuristics' contributions need to be ascertained before proceeding to the implementation phase. Ten examples were selected for the training dataset. These examples, which are natural language requirements specifications, were gathered mainly from database text books.

THE ER-CONVERTER TOOL

Figure 1 depicts the architecture of *ER-Converter*. *ER-Converter* has been implemented using Practical Extraction and Report Language (Perl). The natural language processing involved in the process of translating the database specifications into ER elements is purely based on syntactic analysis.

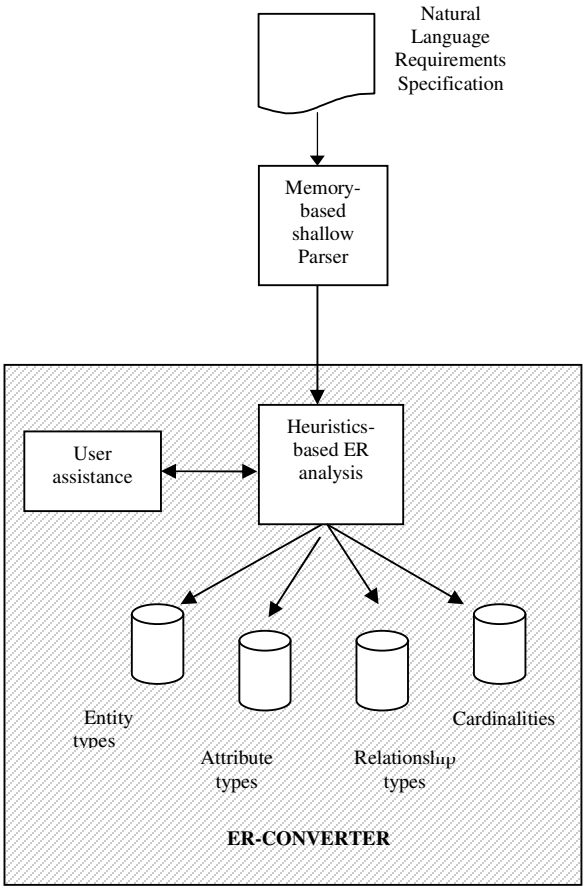

FIGURE 1. ARCHITECTURE OF THE ER-CONVERTER TOOL

The process begins by reading a plain input text file containing a requirements specification of a database problem in English. For this purpose, a parser is required to parse the English sentences to obtain their part-of-speech (POS) tags before further processing. Part of speech tagging assigns each word in an input sentence its proper part of speech such as noun, verb and determiner to reflect the word's syntactic category [1]. The parser used here is Memory-Based Shallow Parser (MBSP) [4,12]. The parsed text is then be fed into ER-Converter to identify suitable data modeling elements from the specification. The task requires several steps to be carried out in order to achieve the desired ER model from the natural language input, each of which is listed as follows:

- Step 1: Part of speech tagging using Memory-based Shallow Parser
- Step 2: Read natural language input text into system
- Step 3: Apply heuristics and assign weights

- Step 4: Human intervention
- Step5: Attachment of attributes to their corresponding entity
- Step 6: Attachment of entities to their corresponding relationship
- Step 7: Attachment of entities to their corresponding cardinality
- Step 8: Produce final result

EVALUATION

The approach in this evaluation uses methods for evaluating Information Extraction systems, primarily Message Understanding Conferences (MUC) [7] evaluations i.e. *recall* and *precision. Recall* is percentage of all the possible correct answers produced by the system. *Precision* is the percentage of answers that are correctly identified by the system. In any system, both precision and recall should be as close to 100% as possible. However, in general, an increase in precision tends to decrease recall and vice versa. In the context of this research, the definition of *recall* and *precision* below are adopted as used by CM-Builder [8] and new measures are defined. Contrary to both precision and recall, all the new measures introduced should be as close to 0% as possible. The measures employed are as follows:

Recall

Recall is the measure of the percentage of information available that is actually found. In this research context, it refers to the amount of the correct information returned by the system. The correct information is then compared with those produced by human analysts or *answer keys*. The following formula is used to calculate recall:

$$Recall = \frac{N_{correct}}{N_{key}} \quad (1)$$

The answer keys or N_{key} is actually the amount of correct information plus the number of missing ones. Thus, the formula is refined as follows:

$$Recall = \frac{N_{correct}}{N_{correct} + N_{part\_correct} + N_{undergenerated} + N_{ask}} \quad (2)$$

Overgenerated

Overgenerated measures how much extra correct information in the system response that is not found in the answer key [8]. This may arise from the use of synonyms in the requirements specification. The following formula is used to measure overgenerated:

$$Overgenerated = \frac{N_{overgenerated}}{N_{correct} + N_{part\_correct} + N_{undergenerated} + N_{ask}} \quad (3)$$

Undergenerated

Undergenerated represents the number of missing correct information that is found in the answer keys but not in the system's response. Thus, $N_{missing}$ below represents the missing items. The following formula (4) is used to calculate undergenerated items:

$$Undergenerated = \frac{N_{undergenerated}}{N_{correct} + N_{part\_correct} + N_{undergenerated} + N_{ask}} \quad (4)$$

Ask_user

Ask_ user represents the number of user assistance requests generated by the system. This user intervention is requested when an item has a low value in its weight and falls between two thresholds. N_{ask} represents *ask user* and the formulas are as follows:

$$Ask\_user = \frac{N_{ask}}{N_{correct} + N_{part\_correct} + N_{undergenerated} + N_{ask}} \quad (5)$$

Unattached

Unattached represents the number of correctly identified ER elements resulting from the system that are not attached to their corresponding items. This inaccuracy need to taken into account as the error will be reflected in the output of the system. $N_{unattach}$ represents this measure. The following formula (6) is used to calculate *unattached*:

$$Unattached = \frac{N_{unattach}}{N_{correct} + N_{part\_correct} + N_{undergenerated} + N_{ask}} \quad (6)$$

Wrongly attached

Wrongly attached measures the numbers of correctly identified ER elements but wrongly attached to other items. This is represented by $N_{wrongattach}$. The following formula (7) is used to calculate this measure:

$$Wrongly\ \ attached = \frac{N_{wrongattach}}{N_{correct} + N_{part\_correct} + N_{undergenerated} + N_{ask}} \quad (7)$$

Precision

Precision is a measure of percentage of correctness of the information produced. It reflects the accuracy of the system in obtaining the correct result. The standard precision formula is as follows:

$$Precision = \frac{N_{correct}}{N_{correct} + N_{incorrect}} \quad (8)$$

In this research, a more detailed formula is used to evaluate the accuracy of the results produced. Apart from

incorrect, other additional measures such as a*sk user*, *undergenerated* and *overgenerated* need to be taken into account for greater accuracy. The following formula (9) is thus defined to calculate precision:

$$Precision = \frac{N_{correct}}{N_{correct} + N_{part\_correct} + N_{ask} + N_{incorrect} + N_{overgenerated}} \quad (9)$$

RESULTS AND DISCUSSION

ER-Converter has been tested using a test dataset which consists of 30 database problems or natural language requirements specification in English. Most of the problems were gathered mainly from database books and past exam papers. Each problem ranges between 50 and 100 words in size. On average, ER-Converter takes 1s to process a database problem which includes processing the tagged input file and generating the ER elements.

ER-Converter achieved a high average *recall* of 90%. The system has successfully produced relevant Entity-Relationship (ER) elements in all of the problems. With a high recall, the heuristics-based system is in better position of applying the corresponding heuristics to the relevant items as compared to the missing ones. 27% of the individual problems or datasets achieved a 100% score in recall. A detailed investigation revealed that all of the missing or *undergenerated* items are either relationships or cardinalities. The undergenerated relationships may due to the fact that verbs are not translated directly as relationships. With respect to the cardinalities, these are mainly due to synonyms and implicit phrases that imply cardinalities. For example, from the phrase "each bus is allocated a particular route", the adjective 'particular' may imply a one-sided cardinality.

In terms of *precision* or correctness of the result produced, ER-Converter scored an average of 85% in the test datasets. The results support that a heuristics-based approach to transform a natural language requirements specification to an ER model can be utilized to aid conceptual modeling in the early stages of database systems development.

ER-Converter has an average of 3% for *overgenerated* items and 6% for *undergenerated* items. The overgeneration are mainly due to synonyms. A detailed investigation revealed that all of the missing or *undergenerated* items are either relationships or cardinalities. The undergenerated relationships may due to the fact that verbs are not translated directly as relationships. For the cardinalities, these are mainly due to synonyms and implicit phrases that imply cardinalities. An interesting result to note is on the user's responses to ER-Converter or referred to as *Ask User* in the evaluation. A user's response is sought when ER-Converter is unsure on whether an ER element is an attribute or an entity. From the evaluation results, it is evident that human intervention in ER-Converter is very minimal with only 2% on average. Although full automation is seen as impossible due to incomplete presentation of knowledge, ambiguities and redundancies [5], this research has shown that it is still possible to provide an almost complete automation with very limited user assistance on the solutions produced. The strength lies in the use of present and newly formed heuristics and the application of their corresponding weights.

RELATION TO OTHER WORK

A comparison in terms of recall and precision is made between ER-Converter and other systems where possible as presented in Table 1. E-R Generator [7] reported that the system was able to identify all the relevant ER relationships and entities in 75% out of 30 database problems that form the test dataset. However, the result was based on only 25% of the total test dataset which were entered interactively by users. The program overgenerated or undergenerated ER entities and relationships in 50% of the cases. No overall results were revealed on the whole test dataset. With ER-Converter, the precision or the accuracy of the system in obtaining the correct result is 85%. However, a direct comparison cannot be made since both systems used different test datasets.

CM-Builder [8] concentrates on building object-oriented conceptual models to be represented in Unified Modelling Language (UML). Though it not comparable in terms of the end results as the system produces object-oriented models and not ER model, the techniques used in the natural language processing and evaluation are similar. Comparing the results with ER-Converter, ER-Converter's performance is well above these figures though a direct comparison is not possible due to the different types of modelling.

System	*Evaluation Results*		
	Recall	*Precision*	*Other*
E-R Generator [6]	75%	-	50%
CM-Builder [8]	73%	66%	62%
ER-Converter	90%	85%	3%

TABLE 1. COMPARISON OF RESULTS WITH RELATED WORK

CONCLUSION AND FUTURE WORK

We have described an approach of generating ER elements from natural language specifications using a heuristics-based system, ER-Converter. The heuristics used are application-domain independent and suitable for small application domains. This study has shown that the

formation of new heuristics in transforming natural language requirements specifications to ER models is supported by the evaluation results. ER-Converter has an average recall of 90% and 85% precision. The contribution made can be applied in areas such as part of the domain model of an intelligent tutoring system, designed to assist in the learning and teaching of databases and other applications of NLP for database design.

REFERENCES

1. Brill, E. 1992. A Simple Rule-Based Part of Speech Tagger. *Proceedings of the Third Conference on Applied Natural Language Processing*, ACL, Trento, Italy : 152-155.
2. Buchholz, E., Cyriaks, H., Dusterhoft, A., Mehlan, H., and B. Thalheim. 1995. Applying a Natural Language Dialogue Tool for Designing Databases. *Proceedings of the First Workshop on Applications of Natural Language to Databases (NLDB'95),* Versailles, France: 119- 133.
3. Chen, P.P. 1983. English Sentence Structure and Entity-Relationship Diagram. *Information Sciences* **1**(1):127-149.
4. Daelemans, W., Zavrel, J., Berck, P. and Gillis, S. 1996. MBT: A memory-based part of speech tagger generator. Ejerhed, E. and Dagan, I. (eds*.), Proc. Of Fourth Workshop on Very Large Corpora*, Philadelphia, USA: 14-27.
5. Eick, C. F. and Lockemann, P.C. 1985. Acquisition of Terminology Knowledge Using Database Design Techniques. *Proceedings ACM SIGMOD Conference*, Austin, USA: 84-94.
6. Gomez, F., Segami, C. and Delaune, C. 1999. A system for the semiautomatic generation of E-R models from natural language specifications. *Data and Knowledge Engineering* **29** (1): 57-81.
7. Grishman, R. and Sundheim, B. 1996. Message Understanding Conference-6: A Brief History. *Proceedings of the 16th International Conference on Computational Linguistics*, Copenhagen, Denmark : 466-471.
8. Harmain, H.M. and Gaizauskas, R. 2003. CM-Builder: A Natural Language-Based CASE Tool for Object-Oriented Analysis. *Automated Software Engineering* **10** (2): 157-181.
9. Storey, V.C. and Goldstein, R.C. 1988. A Methodology for Creating user Views in Database Design. *ACM Transactions on Database Systems* **13** (3): 305-338.
10. Tjoa, A.M and Berger, L. 1993. Transformations of Requirements Specifications Expressed in Natural Language into an EER Model. *Proceeding of the 12th International Conference on Approach*, Airlington, Texas, USA: 206-217.
11. Zanakis, S.H. and Evans, J.R. 1981. Heuristic 'Optimization': Why, When and How to use it. *Interfaces* **11**(5): 84-91.
12. Zavrel, J. and Daelemans, W. 1999. Recent Advances in Memory-Based Part-of-Speech-Tagging. *Actas del VI Simposio Internacional de Communicacion Social*, Santiago de Cuba, Cuba :590-597.